PERFORMANCE MEASURE OF DISTRIBUTED REAL-TIME SYSTEMS UNDER STATIC AND DYNAMIC LOAD SHARING SCHEMES

A SYNOPSIS SUBMITTED FOR THE AWARD OF THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

Science

OF

MODY UNIVERSITY OF SCIENCE AND TECHNOLOGY

Supervisor

Dr. Avanish Kumar
Associate Professor & Former Head
Dept. of Mathematical Sciences and
Computer Applications,
Bundelkhand University, Jhansi-284128, UP, India

Co-Supervisor

Dr. Narendra Kumar
Head, Dept of Science, Faculty of Arts, Science & Commerce
Mody University of Science and Technology,
Lakshmangarh-332311, Sikar, Rajasthan, India

Submitted By

Urmani Kaushal
Enrollment No. 090743

October 2014

"PERFORMANCE MEASURE OF DISTRIBUTED REAL-TIME SYSTEMSUNDER STATIC AND DYNAMIC LOAD SHARING SCHEMES"

1 INTRODUCTION

The availability of inexpensive high performance processors and memory chips has made it attractive to use Distributed Computing Systems for real time applications[1]. Distributed Real-Time Systems are characterized by their requirement that the execution of their computational tasks must be not only logically correct but also completed within time; otherwise, catastrophe may ensue[2]. The term "Distributed Real-Time System [DRTS]" is used to describe whenever there are several computers interconnected in some fashion so that a program or procedure running system terms with multiple processors.In a recently published literature a DRTS has been defined as a system consisting of many heterogeneous processors with different processing capabilities connected by two way communication links and having their own resources/ buffers[3].

However, the term has different meanings to different systems because processors can be interconnected in many ways for various reasons. In the most general form, the word distribution implies that the processors are in geographically separate locations. Occasionally, the term is also applied to an operation using multiple mini-computers, connected with each other and are connected through satellite. A user-oriented definition[4]of distributed computing is "Multiple Computers, utilized cooperatively to solve problems".

1.1 Distributed Real-Time system

Distributed real-time system: A distributed real-time system is a set of real-time systems (nodes) that interact with each other in order to fulfil common task. The nodes are connected by a "real-time communication network"[5].

1.2 Taxonomy Load Distribution Policies

Distributed system provides high performance environments that are competent enough to provide enormous processing capacity [6]. This capacity can be achieved only by making effective and efficient use of resources with the deployment of appropriate load distribution schemes (load balancing or load sharing) and resource allocation algorithms. Maximization of overall system performance is the objective of using an optimality criterion. Computing capacity of a distributed system would not be in proportionate with the number of nodes or processors involved in it [7].

While allocation of resources, the distribution policies should be adopted in such a manner that the processors available in the system neither be overloaded nor stay idle. In the distribution of the processes/tasks, the inter task communication cost incurred should be reduced.

Theory of load distribution classifies the load distributing algorithms as load balancing and load sharing algorithms. The load balancing algorithms equalize the loads at all the processors of the system. It transfers tasks at very higher rate than load sharing algorithms. Load balancing isthe idea of migrating processes across the system from hosts with high loads to hosts with lower loads. It is the ability to split the load toward the same destination. The motivation for load balancing is to reduce theaverage completion time of processes and improvethe utilization of the processors[3, 8].

In load sharing, the tasks are transferred by taking all appropriate decisions. Here once the task has been assigned to the particular node, it will not be reallocated. In load sharing scheme no processor should remain idle while others are heavily loaded in the system[9, 10]. Load sharing is the ability to distribute outgoing traffic over multiple paths. Heavily loaded nodes distribute the workload to lightly loaded nodes under load sharing policies. Performance of distributed systems can be improved by using loadsharing policies/schemes while resource allocation. Various load sharing models has been proposed and implemented in the literature [1, 7, 9, 11].

Load sharing policies may be either static or dynamic. Staticpolicies use only information about the average behaviour of thesystem; transfer decisions are independent of the actual currentsystem state. Static policies may be either deterministic or probabilistic[12]. Dynamic policies, on the other hand,make their load distribution decisions based on the current system state. Various studies have

considered sender-initiated and receiver-initiated load sharing policies as dynamic load sharing policies [13].

1.2.1 Load Balancing vs. Load sharing

The load balancing attempts to equalize the workload on all the available nodes of the system, which increases the overhead of gathering state information. This job of information is very large in distributed system having a large number of nodes. Because a load balancing algorithm transfers tasks at a higher rate than a load sharing, the higher overhead incurred by the load balancing algorithm may outweigh this potential performance improvement. Task transfers are not instantaneous because of communication delays and delays that occur during the collection of task state [14]. Furthermore, load balancing is not achievable in the strictest sense as the number of tasks is always inconsistent over the node and the chronological unbalance among the nodes exists at every moment. For proper resource utilization, it is not necessary to balance the load on all the nodes. Rather, it is necessary and sufficient to prevent the nodes from being idle while some other nodes are heavily loaded [15].

1.3 Task Allocation

Task-to-processor allocation can be dynamic or static. Indynamic allocation, tasks are assigned on the fly, dependingon the system's state.With static allocation, it is assumed thatthe programmer has information about the system's tasksand processors, and that the system makes task assignmentsbefore task execution. The main goal in static task-to-processorassignment is to assign an equal load to all processors andreduce their interaction overhead.

The broad-spectrum problem of task allocation to machines/processors/nodes in DRTS suite has been shown to be NP-complete. Verities of heuristics were developed to perform the task allocations [16].

Task allocation is a restricted case of task scheduling where the tasks have no precedence relations with others and the priority order of execution is ignored[17]. Task allocation problem in DRTS deals with finding appropriate assignment of tasks to available nodes in the system so that the system performance can be maximized. A large body of literature exists for providing the solution and models of task allocation. The solution provided in task assignment algorithms differs in computing system

characteristics, their objectives, and the model of computation. According to the literature [18, 19, 20, 21] the solution methods are categorized as graph theoretic, mathematical programming, state-space search, probabilistic and randomized optimization methods.

2 REVIEW OF LITERATURE

The task allocation problem in a DRTS may either be static or dynamic. If the allocation is static the information regarding over all costs a hardware configuration is known beforehand. When a task is assign to a particular processor, it stays on that processor during its lifetime. The problem of assigning m tasks to n processors, where $m > n$ as normally is the case, in the real life DRTS, is a complex one. The complexity may be reduced by an efficient task partitioning strategy [22]. For important critical applications, the operations in a DRTS must take place with same degree of assurance, which can be achieved by incorporating reliability at various levels of operation. An algorithm for fast allocation in distributed processing environment has been suggested in[23]. In this algorithm author's suggested the matrix reduction methods. Models presented in[24, 25, 26, 27] suggested efficient algorithm for tasks allocations in distributed computing system.

Previous parallel computation models for task allocation have usually included a communication cost, representing the communication delay, associated with the sending of data between tasks which are located on different processors. Early work on this problem used graph theoretic techniques such as network flow and/ or enumeration techniques[28, 29].

Another potential problem related to DRTS is load balancing. In these systems, if some hosts remain idle while others are extremely busy, system performance will be affected drastically. To prevent this load balancing is often used to distribute the jobs and improve performance measures such as the mean response time (MRT). The design of such load balancing algorithms, in general considers several influencing factors, for instance the underlying processors topology, communication network bandwidth, job arrival rates at each processor in the system. Load balancing schemes can be classified as either static or dynamic. A dynamic algorithm[30, 31, 32, 33, 34] makes its decision according to the status of system, where the status could refer to a certain type of information, such as the number of jobs waiting in the queue, the

current job arrival rate, the job processing rate etc., at each processor. On the other hand, a static algorithm[35, 36, 37, 38, 39] performed by predetermined policy, without considering the current status of the system. Dynamic load balancing algorithms offer the possibility of improving load distribution at the expense of additional communication and computation overheads. In [40, 33], it was pointed out that the overhead of dynamic load balancing schemes may be large, especially for a large heterogeneous distributed system. Hence most of the research work in the literature are focused on centralized dynamic load balancing schemes[33, 41, 42]; in which a Management Station (M-Station) kept checking the system status and scheduled the arriving jobs among the processors by some strategies as such Backfilling, Gang Scheduling, Migration[41] etc. By centralizing the M-Station can handle most of the communication and computation overheads efficiently and improve the performance.

The 'high-low policy' for load balancing is proposed in[43], which is also called double-threshold policy. It uses 'high mark' and 'low mark' threshold values to divide the space of possible load states of a node into the three regions which are as follows:

- Over loaded - above the high mark and low mark values
- Normal loaded - above low mark value and below high-mark value
- Under loaded-below both values

It guarantees for the performance at predefined level. The load incurred in sending and receiving a remote task or process has been considered in the proposed model. The decision of processes transfer or retrieval will be taken only after the proper evaluation of incurred cost in the communication.

Load balancing model for homogeneous distributed system proposed in [44]. In the proposed model processors or nodes are intended to inform each other about its load. Based on this information with each processor, the job dispatcher of the processor decides to transfer few of the tasks or not. Either average arrival rate or the amount of unfinished load could be balanced by this algorithm. Because of the decision making polices of the algorithm, extensive mathematical calculations, and the precise numerical load of each processor high overhead is generated over the network.

A drafting algorithm for task migration proposed in [45], which observed that the load measurement for dynamic load balancing scheme is not required to be communicated by the processors. A system of three level systems used in this proposed model. The levels are namely heavy level, normal level and light level. The lightly loaded processor makes the request for task migration from the heavily loaded candidate processor, which sends a bid for confirmation. The migration will start only after the release of select message by lightly loaded processor. To reduce the number of messages due to load broadcasting at every change in load level, to reduce the communication traffic, piggybacking is recommended. To carry out every possible task migration, algorithm has to broadcast all load messages to every other processor in the system.

A gradient model for load balancing for multiprocessor systems is introduced in [46]. The limitation with this model is that a task may be migrated many times without making progress as the load of the processors in the system.

The pairing policy method proposed in [43] for load balancing, reduce the variance of loads only between pairs of nodes of the system. The proposed model increases the migration delay and IPC costs.

In the bidding policy proposed by [47]to balance the load by bidding the process. Each node plays two roles in the system one is of the manager and other is of contractor. The manager tells the system that it requires migrate the process and the contractor represents node's capability of remote process acceptance. Huge communication overhead and very difficult decision policy for good pricing policy of bidding are two major drawbacks of bidding algorithms.

The solution provided in [48] for balancing the load among the processors can be achieved by the minimization of the execution cost of entire program, which in turn maximize the performance and the algorithm is derived from the results in Markov Decision Theory.

The task allocation model proposed in [17] suggested the use of fuzzy logic in the process of task allocation decision making. Based on incomplete and imprecise information, the proposed heuristic tried to provide the mechanism for task assignment

using fuzzy logic. In the proposed model, the constraints like memory, number of process etc. are not considered.

The task allocation model proposed in [49]introduced a new policy for task allocation, SITA-V (Size Interval Task Assignment with Variable Load). This policy does not balance the load in the system. In the suggested model numerous small tasks are redirected to lightly loaded nodes, while the rare large tasks are directed to heavily loaded hosts. Only this single constraint may increase the load over nodes of the system.

In [12], the author compared receiver-initiated and sender-initiated load sharing policies in various workload. Both the load sharing policies give substantial advantages in performance if system is not attempting any load sharing. If the system is lightly or moderately loaded then sender-initiated policies are preferable. While for high system loads receiver-initiated policies should be preferred. Modification in receiver-initiated policies to transfer only newly-arrived tasks yields unsatisfactory performance.

The adaptive load sharing policies have been explored for locally distributed systems and the complexity level of the policy has also been maintained in the simple analytic model proposed for load sharing by [50]. It is found that the performance had improved dramatically by the use of load sharing policies. The use of "threshold-type" information has showed the benefits over no information or complete information type approach.

The model proposed in [51] has suggested that the load distributing algorithm must be adaptive for being effective at meeting either the load balancing or the load sharing objectives over wide range of constraints.

In [13], centralized single coordinator policy performed well in the absence of contention for the coordinator node for small systems. In this policy, the coordinator may become a bottleneck. It may also cause the fault-tolerance problem. This may affect the performance of the system.

The load balancing attempts to equalize the workload on all the available nodes of the system, which increases the overhead of gathering state information. This job of information is very large in distributed system having a large number of nodes.

Because a load balancing algorithm transfers tasks at a higher rate than a load sharing, the higher overhead incurred by the load balancing algorithm may outweigh this potential performance improvement. Task transfers are not instantaneous because of communication delays and delays that occur during the collection of task state [14]. Furthermore, load balancing is not achievable in the strictest sense as the number of tasks is always inconsistent over the node and the chronological unbalance among the nodes exists at every moment. For proper resource utilization, it is not necessary to balance the load on all the nodes. Rather, it is necessary and sufficient to prevent the nodes from being idle while some other nodes are heavily loaded [15]. (From Chapter One)

In[18], a novel clustering approach has been introduced for minimization of task allocation cost in a distributed system. In this approach the task execution costs as weil as the communication costs between the tasks has been considered. Here two metrics had been used to determine the order of task assignment to processors. Clustering any two tasks will guarantee a saving which is equal to the communication cost that incurs if these tasks are assigned to any two different processors under the homogeneous network model. However, when the network is heterogeneous, the amount of saving will no longer be a constant and it will differ according to the processors to which the tasks are assigned.

The task allocation model proposed in [52] has presented the concept of task duplication. In the illustrated examples in the paper, has trimmed down the communication cost by using the proposed task duplication concept. None of the constraints for task allocation has been considered in the proposed model.

Preemptive (resume) scheduling of cooperative tasks that have been preassigned to a set of processing nodes in a distributed system, when each task is assumed to consist of several modules is discussed. During the course of their execution, the tasks communicate with each other to collectively accomplish a common goal. Such intertask communications lead to precedence constraints between the modules of different tasks. The objective of this scheduling is to minimize the maximum normalized task response time, called the system hazard. Real-time tasks and the precedence constraints among them are expressed in a PERT/CPM form with activity on arc (AOA), called the task graph (TG), in which the dominance relationship

between simultaneously schedulable modules is derived and used to reduce the size of the set of active schedules to be searched for an optimal schedule. Lower-bound costs are estimated, and are used to bound the search. An example of the task scheduling problem and some computational experiences are presented[53]

The technique proposed in [54], Task Cost Matrix of $m \times n$ is used to store the execution cost of m tasks over n processors of the distributed system. This matrix is arranged in ascending order of the execution cost and stored in Arranged Task Cost Matrix of $m \times n$. Non interfacing tasks are stored in row matrix $T_{sel}(1,)$. Maximum Number of Task Allocation ($MNTA$) is calculated by using the following equation

$$MNTA = \frac{Number\ of\ selected\ tasks\ in\ T_{sel}(1,)}{Number\ of\ Processors}\ (in\ round\ figures)$$

By using MNTA tasks is allocated to each processor by using First Come First Serve Policy. This step is repeated until all tasks not been executed. The total execution cost is calculated by adding all execution timings of all allocated tasks.

The technique proposed in [55], has split the completed system into three groups on the basis of their computational capacity namely Very High (VH), High (H), and Low (L). The task allocation has been done on the basis of these groups of the systems. For balancing the load in the system task duplication has been used. Here classification method has been used for the optimal assignment of tasks and duplication has been used for the minimization of communication cost. It is not implemented from the view point of distributed real-time system.

In [56], a Processing Time Matrix $PTM(,)$ of $m \times n$ has been used for storing the processing time of m tasks over n processors. The average of each row of the $PTM(,)$ has been stored in $avg_row()$ linear array along with its corresponding tasks. Then this array is sorted in ascending order and stored in another $avg_row_asc()$. Now the tasks has been clustered on the basis of the communication cost and the number of tasks in a cluster should not exceed the limit by $\left[\frac{m}{n}\right] + 1$. According to the clustering done in the above step $PTM(,)$ and Communication Time Matrix $CTM(,)$ has been modified. For finding the optimal assignment the algorithm proposed in [23] has been used.

In [57], the $m$tasks are divided into k phases which are to be allocated over n processors. The phase wise efficiency of individual processor is given in the form of Execution Cost Matrix $ECM(,)$ of order $m \times n$. The Residing Cost of unprocessed tasks on each processor is stored in Residing Cost Matrix $RCM(,,)$ of $k \times m \times n$. The

ITCCM (,) matrix of *m x k* is used to store inter task communication cost. The relocation cost of the tasks is stored in Reallocation Cost Matrix *RECM (,)* of order *m x k*. For each phase *ERCM (,)* has been obtained by adding *ECM (,,)*and *RCM (,,)*. Each row of *ERCM (,)* has been summed up and stored in *sum_row_asc()* in ascending order. From *ERCM (,,)* first *n* tasks has been selected and stored in *sum_row()* and in *ERCM1 (,)*. This process is repeated for next *n* or less than *n* tasks. In all available sub matrices of *ERCM (,,)* allocation has been made by using the technique proposed in [23]. The Execution Cost, Communication Cost, and Reallocation Cost for each sub matrix have been calculated. The overall optimal cost of the system is calculated by summing up the optimal cost of each phase.

In [58], has tried to optimize the system cost by grouping the tasks having high communication in between. The technique proposed in[59], fragmented task allocation scheme in three major steps first "task selection order", second "average processor load" and the third is "processor selection criteria". The task selection list is generated by sorting the tasks according to increasing order of theirselection order.Tie breaking is done randomly. After selection of the order of task, a next criterion is the average processorload is defined as the sum of execution costs of the tasksbeing assigned to it.Because strictly balanced load distribution may not be possible to achieve, a systemis considered to be balanced if the load on each processor is less or equal to the average load. After getting the processors load tasks are assigned sequentially.

3 RESEARCH GAP

The development and use of real-time system is growing very rapidly. As this trend continues, the constraint of performance will become increasingly important. The performance of DRTS can be improved by task allocation. In the performance improvement of DRTS through optimal task allocation, several fundamental problems in model designing are identified.

Majority of the task assignment models proposed in the literature have used the clustering to reduce inter task communication cost first and then device the mechanism to assign the task clusters to appropriate processor or node. While clustering the execution cost of tasks has not been taken into consideration.

The decision of task allocation over any node depends on available memory capacity of the processing node. The memory capacity of the processing node should

be greater than the total memory required by the tasks which are going to be allocated over it. The above constraints had not been considered in the literature while designing the strategies for task allocation.

The allocation decision of a particular task or process on a particular node depends on the number of modules or tasks already in execution on that particular node i.e. the maximum number of modules or tasks which can be assigned to any processing node must be considered.

Load distribution policies are not framed by using any advance technological concepts like ANN and fuzzy logic.

The model proposed and discussed in the literature, focused only in the minimization of execution and inter-task communication cost rather than considering resource utilization. This may causes the uneven load distribution across the system. This load imbalance occurs has been observed in the models proposed in the literature. It may be unsafe for the system performance in terms of the mean response time of tasks and processor utilization.

4 OBJECTIVES

On the basis of above research gap following objectives have been framed:

4.1 Formulation of models to assign the tasks to processors statically and dynamically in Distributed Real- Time System environment considering the load sharing criteria.

4.2 Development of algorithms to assign the tasks to processors statically in Distributed Real-Time System environment.

4.3 Development of algorithms to assign the tasks to processors dynamically in Distributed Real-Time System environment.

4.4 Improving the performance of the Distributed Real-Time System by implementing fuzzy logic approach in load sharing schemes.

5 METHODOLOGY

To fulfil the research objective existing literature was reviewed. Various tasks allocation models for distributed system were studied and based on the study, various limitations of the models for task allocation in DRTS have been observed.

The parallel application should complete its execution over a DRTS in lesser cost than over a standalone system. The parallel applications running over the DRTS is having *m* number of tasks which are to be allocated on *n* number of nodes of the distributed system for execution. A processor can hold limited number of tasks and memory is also the limit. So these constraints had been taken into consideration in the proposed models.

To overcome the research gap analysed by the review of literature, models and its techniques were designed and simulated to achieve the optimal solution for the assignment of the set of *"m"* tasks of programs over the set of *"n"* processors / nodes (*where, m > n*) of DRTS in the current research work. Based on the proposed model three task assignment algorithms under static load sharing scheme, two algorithm for task allocation under dynamic load sharing scheme, and one task allocation algorithm with the inculcation of fuzzy logic under the consideration of dynamic load sharing scheme have been devised. To simulate the model and the algorithms MATLAB has been used.

In order to allocate the tasks of such program to processors in DRTS, the information about certain attributes such as execution cost, inter task communication cost, memory limit vector, number of processors allowed over the nodes should be in hand before.While obtaining such information is beyond the scope of thethesis. Therefore, for the proposed models this required information has been taken from various research papers.

- **Nature of Research**

  Experimental

- **Data Source**
  - Secondary
  - The required information like execution cost, inter task communication cost, memory limit vector, number of processors allowed over the nodes for the implementation of the models has been taken from various research papers.
- **Techniques**

  k-means clustering

munkres

Fuzzy Logic

Static Load Sharing Schemes

Dynamic Load Sharing Schemes

- **Model Design**
- **Algorithm Design**
  - Static Load Sharing Schemes (Three Algorithms)
  - Dynamic Load Sharing Schemes (Two Algorithms)
  - Fuzzy Logic (One Algorithm)
- **Simulation**

  All the models are simulated by using MATLAB.
- **Performance**

  Performance of the proposed models has been compared with existing approaches.

6 SUMMARY

To fulfill the objectives of this research work, four models have been proposed. Based on the proposed models three algorithms were developed and implemented for the task allocation in DRTS considering the static load sharing scheme. And two algorithms were developed and implemented for task allocation in DRTS under dynamic load sharing scheme. One algorithm was designed and implemented based on fuzzy logic under dynamic load sharing scheme. All the algorithms were implemented in MATLAB.

The first chapter of this thesis is devoted to the introductory background of the Distributed Real-Time System. This chapter provides design of distributed system. Further the designing issues of distributed system were discussed. The issue of resource allocation has been elaborated further. The taxonomy of load distribution in distributed real time system has been discussed.Distributed system provides high performance environments that are competent enough to provide enormous processing capacity.This capacity can be achieved only by making effective and efficient use of resources with the deployment of appropriate load distribution schemes (load balancing or load sharing) and resource allocation algorithms. The comparison of load sharing and load balancing policies has been made. In the thesis, we have followed the load sharing policies for improving the performance of DRTS. Further in this chapter

the components of the load sharing have been elaborated. This chapter also discussed about task assignment algorithms, the assignment problem, mathematical formulation of assignment problem, the Hungarian Algorithm, fuzzy logic and the fuzzy logic toolbox in MATLAB.

In the second chapter review of literature has been done. In the literature it has been find out that by optimal allocation of the parallel application's tasks, the performance of the system can be enhanced. The broad-spectrum problem of task allocation to machines/processors/nodes in DRTS suite has been shown to be NP-complete.Various popular techniques from variety of fields have been incorporated in the models given for task allocation. Techniques like genetic algorithm, fuzzy logic, simulated annealing, graph theoretic technique, opportunistic load balancing, minimum execution time, minimum completion time, min-min, max-min and duplex etc. are most common techniques that have been used in various ways in different proposed model for task allocation. The above common task allocation techniques have been discussed in this chapter. Further the specialized task allocation techniques have also been discussed here. On the basis of the review of literature the research gap has been identified. And based on the research gap the objectives of the thesis have been set.

In the third chapter, the model proposed to minimize the overall processing cost for a distributed system through optimally assigning the tasks of various processors of the DRTS, so that the performance of the distributed system could be upgraded. The proposed model was based on static load sharing scheme and effective clustering mechanism.Astraight forward and efficient procedure for task allocation has been adopted in the model. The clustering of tasks has reduced the processing cost. Finally the same numbers of clusters were formed as the number of processors in DRTS. The complexity of the task allocation algorithm given by Richard et al. [60] and Kapil et al. [61] are $O(n^m)$ *and* $O(m^2n^2)$ respectively. The overall complexity of the proposed algorithm is $O(m^2)$.The complexity is much less than that of other similar algorithms [60][61].

The fourth chapter was aimed to enhance the performance of the DRTS by making optimal utilization of its processors and suitable allocation of tasks.The algorithm designed under the consideration of static load sharing scheme with the goal of minimizing system cost, and maximizing the Processor Utilization. The proposed

model is based on an effective clustering to reduce *ITCC*. To measure the performance of proposed algorithm, the same example, present in Kapil et al. [61] has been solved. In this case, it had shown that the system cost was minimized by 7.69 % and the APU was maximized by 12 %.

The model proposed in fifth chapter, offer an optimal solution by assigning a set of "*m*" tasks of the parallel application to a set of "*n*" processors (where, $m > n$) in a distributed real-time system with the goal to enhance the performance of distributed systemby making optimal utilization of its processors and suitable allocation of tasks.The model proposed here is using static load sharing scheme. The*ITCC* has been reduced by multilevel clustering and the system cost has been minimized by efficient and straightforward manner for optimal task allocation. The proposed model has been implemented the example presented in Bora et al. algorithm[18]. The results obtained by implementing the proposed model has reduced the total system cost in comparison with the cost obtained by Bora et al. algorithm[18], Lo's algorithm[62] and Kopidakis et al. algorithm[63], Yadav et.al.[58]and Kumar et. al.[64] by 27.78%, 29.09%, 31.58%, 25% and 33.33%respectively.

In the sixth chapter, to improve the performance of the system, a new heuristic has been suggested and implemented. The number of modules which can be assigned on the processor is limited and the memory is also having certain limit. So these two constraints have been taken into consideration while designing the model. The dynamic load sharing policy has been used to improve the performance i.e. at the time of assignment, the required constraints must be check and fulfilled. For clustering task k-mean clustering has been used.To measure the performance of proposed model & algorithm, the same example, present in Vidyarthi et al.[65] has been solved. In this case, it showed that the system cost was minimized by 20 % and the APU was maximized by 8.35 %.

In the seventh chapter, the proposed model was based on *k-mean* clustering and effective clustering at the time of assigning the tasksto the nodes. The straightforward and efficient algorithm had been designed to obtain optimal system cost. Before allocation of tasks on the processor, node's memory capacity has been checked and number of tasks allowed by the processor has also been checked. Restricted assignments given by the initial phase of the algorithm has also been considered at time of the allocation. The problem solved by using the proposed model has

minimized total system cost by 3.01205 % in comparison with the model proposed by Urmani et al.[66].

The model proposed in chapter eighth was based on effective two levels clustering and fuzzy logic based approach for checking memory constraint. The task allocation is done using dynamic load sharing scheme. Before allocation of tasks on the processor, node's memory capacity has been checked and the decision has been taken by applying fuzzy logic. Restricted assignments decided by memory availability which was calculated by using fuzzy logic has been considered at time of the allocation. In the solved problem the Average Processor Utilization (AUP) has increased by 19.968 % in comparison with the model proposed in chapter seven.

REFERENCES

[1] C. Hou and K. Shin, "Load sharing with consideration of future task arrivals in heterogeneous distributed real-time systems," *Computers, IEEE Transactions,* vol. 43, no. 9, pp. 1076-1090, 1994.

[2] K. G. Shin, C. Krishna and L. Yann-Hang, "A unified method for evaluating real-time computer controllers and its application," *Automatic Control, IEEE Transactions,* vol. 30, no. 4, pp. 357-366, 1985.

[3] Z. Zeng and B. Veeravalli, "Design and performance evaluation of queue-and-rate-adjustment dynamic load balancing policies for distributed networks," *Computers, IEEE Transactions,* vol. 55, no. 11, pp. 1410-1422, 2006.

[4] D.-T. Peng and K. G. Shin, "Optimal scheduling of cooperative tasks in a distributed system using an enumerative method," *Software Engineering, IEEE Transactions,* vol. 19, no. 3, pp. 253-267, 1993.

[5] H. Kopetz, "REAL-TIME SYSTEMS: Design Principles for Distributed Embedded Applications," *Kluwer Academic Puplishers,* 1997.

[6] T. Lewis and H. El-Rewini, INTRODUCTION TO PARALLEL COMPUTING, Prentice-Hall, Inc., 1992.

[7] K. Y. Kabalan, W. W. Smari and J. Y. Hakimian, "Adaptive Load Sharing in Hetergeneous Systems: Policies, Modifications, and Simulation," *I. J. of SIMULATION,* vol. 3, no. 1-2, pp. 89-100, June 2002.

[8] M. H. Balter and A. B. Downey, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing," Berkeley, California, 1995.

[9] H. D. Karatza and R. C. Hilzer, "Epoch Load Sharing in a Network of Workstations," in *Proceedings of the 34th Annual Simulation Symposium (SS '01) IEEE*, 2001.

[10] U. Kaushal and A. Kumar, "Improving the Performance of DRTS by Optimal Allocation of Multiple Tasks under Dynamic Load Sharing Scheme," *International Journal of Scientific & Engineering Research,* vol. 4, no. 7, pp. 1316-i321, July 2013.

[11] M. Lo and S. P. Dandamudi, "Performance of Hierarchical Load Sharing in Heterogeneous Distributed Systems," in *Int. Conf. Parallel and Distributed Computing Systems*, Dijon, France, 1996.

[12] D. L. Eager, E. D. Lazowska and J. Zahorjan, "A comparison of receiver-initiated and sender-initiated adaptive load sharing," *ACM SIGMETRICS Performance Evaluation Review,* vol. 13, no. 2, pp. 1-3, 1985.

[13] M. Lo and S. P. Dandamudi, "Performance of Hierarchical Load Sharing in Heterogeneous Distributed Systems," in *Proc. Int. Conf. Parallel and Distributed Computing Systems*, Dijon, France, 1996.

[14] M. Singhal and N. G. Shivaratri, Advanced Concepts in Operating Systems, 1st ed., C. Liu, Ed., New Delhi: Tata McGraw-Hill Education Private Limited, 2011.

[15] P. K. Sinha, Distributed Operating Systems Concepts and Design, New Delhi: Prentic Hall of India Private Limited, 2003.

[16] T. D. Braun, H. J. Siegel, N. Beck, L. Boloni, M. M., A. I. Reuther, J. Robertson, M. Theys and B. Yao, "A taxonomy for describing matching and scheduling heuristics for mixed-machineh eterogeeterogeneous," in *IEEE Ibrkshop on*

Advance.3 in Parallel and Distributed Systems (APADS '98), 1998.

[17] S. Salleh, B. Sanugi and H. Jamaluddin, "Task Allocation In A Multiprocessor System Using Fuzzy Logic," *Jurnal Teknologi,* vol. 25, no. 1, pp. 69-79, 1996.

[18] B. Ucara, C. Aykanata, K. Kayaa and M. Ikincib, "Task Assignment in heterogeneous computing system," *J. Parallel Distrib. Comput.,* vol. 66, pp. 32-46, 2006.

[19] M. Dhodhi, I. Ahmad, A. Yatama and I. Ahamad, "An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems," *J. Parallel Distrib. Comput.,* vol. 62, no. 9, p. 1338–1361, 2002.

[20] K. Efe, "Heuristic models of task assignment scheduling in distributed Systems," *IEEE Comput.,* vol. 15, no. 6, p. 50–56, 1982.

[21] A. Khandelwal, "Task Allocation for Distributed Computing Systems: From Past to Present," *International Journal of Advanced Research in Computer Science,* vol. 3, no. 2, pp. 282-287, April 2012.

[22] P. K. Yadav and A. Kumar, "An Efficient Static Approach for Allocation through Reliability Optimization in Distributed Systems," in *International conference on Operations Research for Development*, Chennai, 2002.

[23] A. Kumar, M. Singh and P. Yadav, "A Fast Algorithm for Allocating Tasks in Distributed Processing System," in *Proc. of the 30th Annual Convention of CSI*, Hyderabad, India, 1995.

[24] A. Kumar, M. P. Singh and P. Yadav, "An Efficient Algorithm for Allocating Tasks to Processors in a Distributed System," in *Proc. of the 19th National system conference, SSI*, Coimbatore, India, 1995.

[25] V. Kumar, P. Yadav and K. Bhatia, "Optimal Task Allocation in Distributed Systems owing to Inter Tasks Communication Effects," in *Proc. of the 33rd Annual convention of system society of India*, New Delhi, India, 1998.

[26] W. W. Chu, "Optimal file allocation in a multiple computer system," *IEEE Transactions on Computer,* vol. 100, no. 10, pp. 885-889, 1969.

[27] P.Y. Richard MA, E. Y. S. Lee and M. Tsuchiya, "A Task Allocation Model for Distributed Computing Systems," *IEEE Transactions on Computers,* vol. C 31, no. 1, pp. 41-47, January 1982.

[28] O. El-Dissouki and W. Huen, "Distributed enumeration on network computers," *IEEE Trans. on Computers,* vol. 29, no. 9, pp. 818-825, 1980.

[29] H. S. Stone, "Multiprocessor scheduling with the aid of network flow algorithms," *IEEE Trans. On Software Engineering,* vol. 3, no. 1, pp. 85-93, 1977.

[30] D. J. Evans and W. U. N. Butt, "Dynamic load balancing using task-transfer probabilities," *Journal of Parallel Computing,* vol. 19, no. 8, pp. 897-916, 1993.

[31] C. Walshaw and M. Berzins, "Dynamic load-balancing for PDE solvers on adaptive unstructured meshes," *Concurrency: Practice and Experience,* vol. 7, no. 1, pp. 17-28, 1995.

[32] J. Watts and S. Taylor, "A practical approach to dynamic load balancing," *IEEE Trans. on Parallel and Distributed Systems,* vol. 9, no. 2, pp. 235-248, 1998.

[33] Y. Zhang, H. Kameda and K. Shimizu, "Adaptive bidding load balancing algorithms in heterogeneous distributed systems," in *Proceedings of the Second International Workshop on Modeling, Analysis, and Simulation of Computer and*

Telecommunication Systems MASCOTS'94 IEEE, 1994, January.

[34] L. Anand, D. Ghose and V. Mani, "ELISA: an estimated load information scheduling algorithm for distributed computing systems," *Computers & Mathematics with Applications,* vol. 37, no. 8, pp. 57-85, 1999.

[35] D. Grosu and A. T. Chronopoulos, "A game-theoretic model and algorithm for load balancing in distributed systems," in *In Parallel and Distributed Processing Symposium, International*, 2002.

[36] J. Li and H. Kameda, "Load balancing problems for multiclass jobs in distributed/parallel computer systems," *IEEE Transactions on Computers,* vol. 47, no. 3, pp. 322-332, 1998.

[37] A. N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," *Journal of the ACM (JACM),* vol. 32, no. 2, pp. 445-465, 1985.

[38] K. I. M. Chonggun and H. Kameda, "Optimal static load balancing of multi-class jobs in a distributed computer system," in *Proc. of 10th Intl. Conf. Distributed Computing Systems*, 1990.

[39] Z. Zeng and B. Veeravalli, "Design and analysis of a non-preemptive decentralized load balancing algorithm for multi-class jobs in distributed networks," *Computer Communications,* vol. 27, no. 7, pp. 679-693, 2004.

[40] Y. Zhang, K. Hakozaki, H. Kameda and K. Shimizu, "A Performance Comparison of Adaptive and Static Load Balancing in Heterogeneous Distributed Systems," in *Proceedings of the 28th Annual Symposium in Simulation*, 1995.

[41] Y. Zhang, H. Franke, J. Moreira and A. Sivasubramaniam, "An integrated approach to parallel scheduling using gang-scheduling, backfilling, and migration," *IEEE Transactions on Parallel and Distributed Systems,* vol. 14, no. 3, pp. 236-247, 2003.

[42] E. Choi, "Performance test and analysis for an adaptive load balancing mechanism on distributed server cluster systems," *Future Generation Computer Systems,* vol. 20, no. 2, pp. 237-247, 2004.

[43] L. Cova and R. Aloson, "Sharing Jobs among Independently owned Processors," in *Proceedings of the 8th International Conference Systems, IEEE*, New York, 1988.

[44] C. Gao, J. Liu and M. Railey, "Load Balancing Algorithms in Homogeneous Distributed Systems," in *Proceedings of the 1984 International Conference on Parallel Processing*, Bocaraton,Fl, 1984.

[45] L. Ni, C. Xu and T. Gendreau, "A Distributed Drafting Algorithm for Load Balancing," *IEEE Trans. on Software Engg.,* Vols. SE-13, no. 10, pp. 1153-1161, October 1985.

[46] F. Lin and R. Keller, "The Gradient Model Load Balancing Method," *IEEE Trans. on Software Engg.,* Vols. SE-13, no. 1, pp. 32-38, January 1988.

[47] R. Bryant and R. Finkel, "A Stable distributed Scheduling Algorithm," in *Proceedings of the 2nd International Conference on Distributed Computing Systems, IEEE*, New York, 1981.

[48] C. Waldspurger, T. Hogg, A. Huberman, J. Kephart and W. Stonetta, "Spawn: A Distributed Computational Economy," *IEEE Trans. on Software Engg.,* vol. 18, no. 2, pp. 103-117, 1992.

[49] M. E. Crovella, M. Harchol-Balter and C. D. Murta, "Task Assignment in A Distributed System (Extended Abstract): Improving Performance By Unbalancing Load," *ACM SIGMETRICS Performance Evaluation Review,* vol. 26, no. 1, pp. 268-269, 1998.

[50] D. L. Eager, E. D. Lazowska and J. Zahorjan, "Adaptive load sharing in homogeneous distributed systems," *IEEE Transactions on Software Engineering,* Vols. SE-12, no. 5, pp. 662-675, 1986.

[51] P. Krueger and M. Livny, "Load balancing, load sharing and performance in distributed systems," University of Wisconsin-Madison, Madison, 1987.

[52] R. Sharma and Nitin, "Duplication with Task Assignment in Mesh Distributed System," in *World Congress on Information and Communication Technologies*, 2011.

[53] D. T. Peng and K. G. Shin, "Optimal scheduling of cooperative tasks in a distributed system using an enumerative method," *IEEE Transactions on Software Engineering,* vol. 19, no. 3, pp. 253-267, 1993.

[54] K. Govil, A. Rastogi and A. Kumar, "An Efficient Algorithm for Task Allocation in Distributed Networking Environment," in *National Conference on Computing For Nation Development,INDIACom-2011*, New Delhi, 2011.

[55] R. Sharma and Nitin, "Duplication with Task Assignment in Mesh Distributed System," in *World Congress on Information and Communication Technologies*, 2011.

[56] K. Govil and A. Kumar, "A Modified and Efficient Algorithm for Static Task Assignment in Distributed Processing Environment," *International Journal of Computer Applications,* vol. 23, no. 8, pp. 1-5, 2011.

[57] K. Govil, "A Smart Algorithm for Dynamic Task Allocation for Distributed Processing Environment," *International Journal of Computer Applications,* vol. 28, no. 2, pp. 13-19, August 2011.

[58] P. Yadav, M. Singh and K. Sharma, "An Optimal Task Allocation Model for System Cost Analysis in Hetrogeneous Distributed Computing Systems: A Heuristic Approach," *International Journal of Computer Applications,* vol. 28, no. 4, pp. 30-37, August 2011.

[59] P. K. Yadav, M. Singh and K. Sharma, "Task Allocation Model for Reliability and Cost Optimization in Distributed Computing System," *International Journal of Modeling, Simulation, and Scientific Computing,* vol. 2, no. 2, pp. 131-149, 2011.

[60] P. Y. Richard, M. A. Edward, Y. Lee and M. Tsuchiya, "A Task Allocaion Model for Distributed Computing System," *IEEE Trans. on Computers,* Vols. C-31, no. 1, pp. 41-47, 1982.

[61] K. Govil and A. Kumar, "A Modified and Efficient Algorithm for Static Task Assignment in Distributed Processing Environment," *International Journal of Computer Applications,* vol. 23, no. 8, pp. 1-5, June 2011.

[62] V. Lo, "Heuristic algorithms for task assighment in distributed system," *IEEE Trans. Comput.,* vol. 37, no. 11, pp. 1384-1397, 1988.

[63] Y. Kopidakis, M. Lamari and V. Zissimopoulos, "Task assignment problem:two new heuristic algorithms," *J. Parallel Distrib. Comput.,* vol. 42, no. 1, pp. 21-29, 1997.

[64] H. Kumar, "A Task Allocation Model for Distributed Data Network," *Journal of*

Mathematical Sciences, vol. 1, no. 4, pp. 379-392, 2006.

[65] D. Vidyarthi and A. Tripathi, "Maximizing Reliability of Distributed Computing Systems with Task Allocation using Simple Genetic Algorithm," *J. of Systems Architecture,* vol. 47, pp. 549-554, 2001.

[66] U. Kaushal and A. Kumar, "Improving the Performance of DRTS by Optimal Allocation of Multiple Tasks under Dynamic Load Sharing Scheme," *International Journal of Scientific & Engineering Research,* vol. 4, no. 6, pp. 1316-1321, June 2013.